Bachelor Thesis
-
Analysis and simplification of architectural floor plans

Principals:	PlanFabrik GmbH
Authors:	Alexander Wyss, Florian Bruggisser
Supervising Prof.:	Simon Schubiger, Stefan Arisona
University:	FHNW Technik
Email:	alexander.wyss@students.fhnw.ch
	florian.bruggisser@students.fhnw.ch

January 23, 2017

1 Abstract

This work's main topic is the analysation of floor plans to detect and mark areas where different types of floor heatings will be installed. The idea is to automate this process for simple floor plans and provide additional tools to correct possible errors in more complicated plans.

This work will be split into sever parts. The first part will describe several algorithms used for preprocessing the image to remove any noise. Additionally, the algorithms for room detection and object detection will be explained.

The second part will contain the whole project management and implementation. This includes for example a timetable of our work, the class structure and much more information concerning the project setup and execution.

Last but not least, we will present and dicuss the results of our work. This also includes features that may be built in the future and the limitations of our work.

Contents

2 Introduction

2.1 Starting position

The company PlanFabrik GmbH creates floor plans for house technology. In a first step the plans are analysed and enhanced through an employee. He adds more information to the plan, like room polygons or he marks areas where house technology can be installed and where not. Those areas are defined by different features. For example, usually the area close to windows has a higher density of pipes emitting heat to the room. This is to equal out the heat loss that usually occurs on windows.
The process of drawing those areas is currently tool supported but still takes a lot of time, equal to the rising amount of rooms the employee has to analyse. The idea is to create a more automated system which does require only a small amount of user input.

2.2 Problem description

To manually create all the room-areas, it takes the user a lot of time. This is due to the fact that he has to click every corner of each room to make the room-enclosing polygon. The fact that the current tool is not too precise, which means that errors in selecting the exact corner exist regularly. The user then has to rearrange the created corner to fit the exact corner in the image. This process of selecting all the corners by hand is highly inefficient, especially for floor plans that have several hundred or even thousands of corners in it.
An automated solution would provide a faster and more comfortable solution for the client. Additionally, the software may provide additional info that is needed. It can calculate the size of each room directly, which would otherwise have to be calculated by hand.

2.3 Goals

The idea of theory of this bachelor thesis is to compare different algorithms which can automatically analyse floor plans and draw the polygons for the rooms. The goal in the end is to show and explain a way to solve the problem and decide which algorithms are used best. All of this will help us create a software that can do the automated analysis for basic floor plans. More complicated floor plans may hold problems that our algorithms can not find, those are supposed to be solved manually with an editor.
In the end, the program is supposed to create an output-file with the information of the areas found which is exportable to the existing software that currently handles the following processes.

3 Theory

3.1 Information Segmentation

Erosion and Dilation

In our project we need erosion to strengthen small features that we want to be highlighted as well as to get rid of small details in the picture itself. We use it in early preprocessing. It helps together with the dilation to clean up pictures to further process them without too many interfering objects in the image. The main usage is to extract an image that features all the walls and has as few as possible other lines on it. [1]
For our project we use the binary grayscale erosion. The erosion uses the following principle to work.

$$A \ominus B = \{z \in E | B \subseteq A\}$$

$$where : B_z = \{b + z | b \in B\} with \forall z \in E$$

A is a binary image in the Euclidean space or an integer grid. The erosion is done with the structuring element B on the image A. The structuring element has to be a subset or equal to A, otherwise there will be no erosion. The mask B will be applied to all pixels possible, if the mask fits on the center pixel of A, the value will be retained. Otherwise it will get deleted. This means that only when B is completely contained in A, values of pixels are retained.
Binary dilation uses the exact same process. The only difference compared to the erosion is that the deletion of the pixel will not be setting pixel values to black (0) but to white (1). Both of those processes are inherently the same and can be summed up under the term morphological operation. These are altering the image with a mask, as we did here for erosion and dilation.

Distance transformation

The distance transformation is as well as the erosion and dilation a morphological operation. In our project it serves the purpose to find bassins for the watershed algorithm. This is used to define the foreground picture. We will explain this further in the wathershed discussion. It is used after preprocessing to find the center of what is supposed to be rooms. This algorithm is only used in combination with the watershed.

The algorithm that openCV uses is called cvDistTransform and uses an euclidean distance to calculate the output image. The input for this transformation is a binary image

$$I(u, v) = I(x)$$

which will be separated into a foreground and background image.

$$FG(I) = x | I(x) = 1$$

$$BG(I) = x|I(x) = 0$$

The formula for the distance transformation of I, D(x) itself is defined as:

$$D(x) := \min_{x' \in FG(I)} dist(x, x')$$

It applies to any pixel x = (u,v) of the input image. If the pixel is part of the foreground image FG(I) then the result of the transformation D(x) is zero. As said above, the distance formula dist(x,x') is the euclidean distance (also called L_2-Norm) between two pixels.

$$dist(x, x') = \|x - x'\| = \sqrt{(u - u')^2 + (v - v')^2}$$

Since the exact calculation with the euclidean distance takes a lot of computational power openCV uses the Chamfer-Algorithm instead. This algorithm runs a two masks over the image. The first mask M_L is run over in a diagonal manner over the image starting in the top left corner of the image. The second mask M_R does the same in reverse starting at the bottom right corner. Those masks are a matrix that represent the euclidean distance between the center and the pixels around it. Each mask only changes the pixels in the direction they are run through the image [1]. As a result, the masks look similar to the following ones:

$$M_L = \begin{bmatrix} . & . & . \\ . & x & m_1 \\ m_2 & m_1 & m_2 \end{bmatrix}$$

$$M_R = \begin{bmatrix} m2 & m1 & m2 \\ m1 & x & . \\ . & . & . \end{bmatrix}$$

This all leads to the resulting effect shown in the image below.

Source: http://stackoverflow.com/questions/7426136/fastest-available-algorithm-for-distance-transform

Make image of distance transformation before and after

3.2 Structural Analysis

Orientated FAST and Rotated BRIEF

Write why SURF not used: License

Cascade training

3.3 Semantic Analysis

Hough transformation

Watershed

The watershed-algorithm in our project is used for segmentation of the different rooms. It can find rooms indifferent of its shape.

The algorithm is processed on a grayscale image on which the color intensity is analogous to the height in a height map. The watershed in use does flooding. The idea is to place a water source in each regional minimum and flood the entire relief. It will stop if it meets a different watersource or an impassable barrier.

4 Implementation

4.1 Experiment

Description of Test-set

4.2 Parameter Finding

Parameter finding and description

4.3 Information Segmentation

Noise Removal (Morphological)

4.4 Structural Analysis

Distance Transform, Why not Contour Detection

4.5 Semantic Analysis

Object Detection: TM, Hough Lines, ORB

Cascading Classifier: To few samples, Cat images prove, more positive, thickening, polydp, positive negative ratio, more specific negative samples

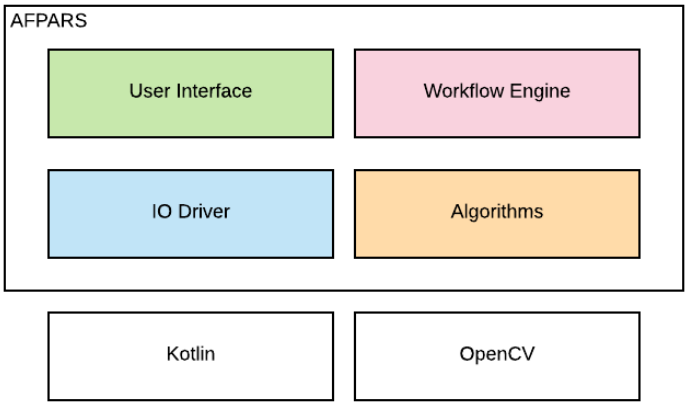


Figure 1: AFPARS software architecture.

5 Software Architecture

This chapter describes the solution developed during our work. At the beginning the software architecture will be explained, followed by the partial steps and experiments which were needed to find the solution.

The software architecture has to fulfil multiple requirements for this project. It should be easily extendable with new algorithms and ui extensions. The input and output format of the application should be independent from the algorithms and the ui to use different file types like bitmap images or Dxf/DWG formats. The algorithms of the application should be link together as workflows which then can be executed by an engine in parallel.

Requirement - Solution

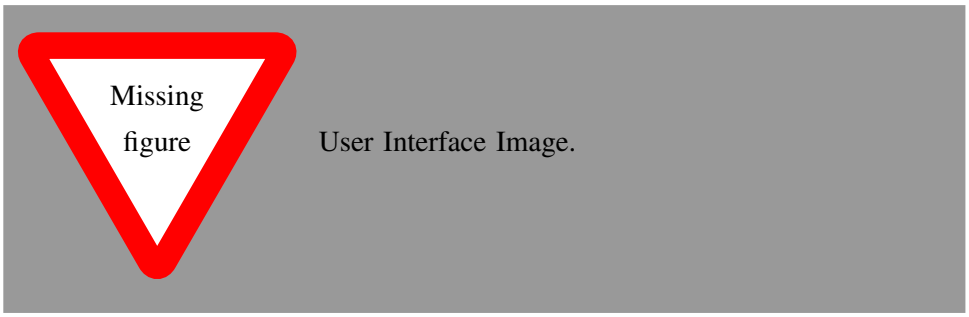
5.1 Meta Format

To support the different input and output formats the architecture uses a meta format for the floor plans called AFImage.

5.2 Algorithm

5.3 Workflow

5.4 User Interface



User Interface Image.

Why it looks like that. (Andere Software)

6 Result

7 Conclusion

8 Developer Guide

8.1 Add Algorithm

8.2 Edit UI

Bibliography

[1] W. Burger and M. J. Burge, *Digital image processing: an algorithmic introduction using Java*. Springer, 2016.

List of Figures